情報処理学会論文誌 Vol.59 No.1 1–2 (Jan. 2018)

論文タイトル

吉田 昂太

受付日 xxxx年0月0日, 採録日 xxxx年0月0日

Kota Yoshida

Received: xx 0, xxxx, Accepted: xx 0, xxxx

1. 概要

現在の多くのプロセッサは投機的実行を悪用する Spectre 攻撃に対して脆弱である。これらの攻撃に対処するソフトウェアベースの方法として、プログラム中から Spectre 攻撃に対して脆弱なコード辺 (Spectre ガジェット) を特定し、部分的に投機的実行を抑制する方法がある。既存研究では Spectre ガジェットを検出する方法として記号実行を用いる手法が提案されているが、通常の実行パスと投機的実行パスの両方を探索する必要があるため、探索する状態空間が非常に多くなり複雑なプログラムに対してスケールしない問題がある。本論文では、Spectre ガジェットの検出確率が低い投機的な状態の探索を避けることで、記号実行のスケーラビリティを向上させる手法を提案する。また、記号実行において探索されなかった投機的状態はファジングを用いて探索することで、スケーラビリティと精度の両立を目指す。

2. はじめに

3. 背景

3.1 Speculative execution

3.2 Transient Execution Attack

一時実行攻撃とは、CPU の投機的実行によって一時的に実行される命令がマイクロアーキテクチャに痕跡を残すことを利用する攻撃法である。本来、CPU は誤った投機的実行が行われた場合、その結果はマイクロアーキテクチャに反映されず、パイプラインはフラッシュされる。しかし、キャッシュなどの一部のマイクロアーキテクチャの状態はパフォーマンスの観点からそのまま維持される。攻撃者はこれを Prime+Probe [1] や Flush+Reload [2] でサイドチャネルを経由して秘密情報を読み取る。一時実行攻撃は 2018 年に Spectre 攻撃 [3] と Meltdown 攻撃 [4] が初めて明らかにされて以来、様々な CPU を標的とした、多数の新しい一時実行攻撃が発見されてきた。これらの攻撃は大きく分けて Spectre 型と Meltdown 型に分類される [5]。Spectre 型は分岐予測ミスに続く一時的な命令を悪用する。一方で, Meltdown 型はフォールトを発生させる命令に続く一時的な命令を悪用する。一時実行攻撃では、通常キャッシュを利用して漏洩したデータを読み取るが、他のサイドチャネルが利用される場合もある [6, 7]。

3.3 Spectre attack

3.4 記号実行

3.5 ファジング

ファジングは、ソフトウェアの欠陥や脆弱性を検出することを目的としたテスト手法である。ファジングは多数のテストケースを対象ソフトウェアへの入力として生成し、

その実行結果を観測することでバグや脆弱性を検出する。単純にランダムにテストケースを生成すると入力空間が膨大になり非効率的であるため、多くのファジングツールは冗長なテストケースやバグを起こす可能性の低いテストケースの生成を回避する手法を用いている。ファジングに関するほとんどの研究はソフトウェアをテストすることを目的としているが、近年、ハードウェアを対象としたファジングの研究が活発になっている [8–10]。Osiris [8] はタイミングベースのサイドチャネル攻撃の特性を利用することで、ターゲット CPU のタイミングベースのサイドチャネルを自動で検出することができる。Transynther [9] は投機的実行を引き起こすことが知られているコードを変更し、マイクロアーキテクチャデータサンプリング (MDS) 攻撃の亜種を検出できるツールである。Revizor [10] は商用のブラックボックス CPU における投機的実行による脆弱性を検出するツールである。

4. KLEESpectre

5. 問題設定

モチベ例について説明
SpecFuzz の結果を引用

6. 提案手法

7. 実装

8. 評価

9. 関連研究

10. 結論

参考文献

[1] Colin Percival. Cache missing for fun and profit, 2005.

[2] Yuval Yarom and Katrina Falkner. {FLUSH+ RELOAD}: A high resolution, low noise, l3 cache {Side-Channel} attack. In *23rd USENIX security symposium (USENIX security 14)*, pp. 719–732, 2014.

[3] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. Spectre attacks: Exploiting speculative execution. *Communications of the ACM*, Vol. 63, No. 7, pp. 93–101, 2020.

[4] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, et al. Meltdown: Reading kernel memory from user space. *Communications of the ACM*, Vol. 63, No. 6, pp. 46–56, 2020.

[5] Claudio Canella, Jo Van Bulck, Michael Schwarz, Moritz Lipp, Benjamin Von Berg, Philipp Ortner, Frank Piessens, Dmitry Evtyushkin, and Daniel Gruss. A systematic evaluation of transient execution attacks and defenses. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 249–266, 2019.

[6] Atri Bhattacharyya, Alexandra Sandulescu, Matthias Neugschwandtner, Alessandro Sorniotti, Babak Falsafi, Mathias Payer, and Anil Kurmus. Smotherspectre: exploiting speculative execution through port contention. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 785–800, 2019.

[7] Michael Schwarz, Claudio Canella, Lukas Giner, and Daniel Gruss. Store-to-leak forwarding: leaking data on meltdown-resistant cpus (updated and extended version). *arXiv preprint arXiv:1905.05725*, 2019.

[8] Daniel Weber, Ahmad Ibrahim, Hamed Nemati, Michael Schwarz, and Christian Rossow. Osiris: Automated discovery of microarchitectural side channels. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1415–1432, 2021.

[9] Daniel Moghimi, Moritz Lipp, Berk Sunar, and Michael Schwarz. Medusa: Microarchitectural data leakage via automated attack synthesis. In *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1427–1444, 2020.

[10] Oleksii Oleksenko, Marco Guarnieri, Boris Köpf, and Mark Silberstein. Hide and seek with spectres: Efficient discovery of speculative information leaks with random testing. *arXiv preprint arXiv:2301.07642*, 2023.